Appendix A:

Netscaler / Native linux LB -

Because of the MAC forwarding method we were forced to use in the new versions of IBM load balancer is the option for TCP listener application to use TCP load balancer was gone for one of our critical application.

Can't load balance TCP. So different load balancer had to be used, preferably inside the system Z.

Netscaler has built an HA and two Netscalers can work together to provide HA within load balancer and load balance the TCP application and is a very powerful but pricey load balancing solution for the enterprise. Netscaler has checks for health check of real servers and easy web interface to manage supports VLANs and many advanced features for routing traffic.

Netscaler can be an HTTP load balancer/HTTPS SSL offload load balancer/and TCP service load balancer.

We needed it to do TCP load balancing for passport application.
we run our home grown Java app which listens on TCP port 9666 over NFS shared data on two zLINUX servers.

Setup was easy:
Implement a Netscaler VPX virtual machine in VMware.

- Extract virtual image
- Install license
- Configure Mapped IP, Netscaler IP etc
- Open required firewall ports to real server
- Setup DNS to point the URL to Netscaler IP
- Setup Netscaler service monitoring to monitor real servers.

Setup real server config in Netscaler to load balance TCP port 9666
Setup the service group in Netscaler

The plan is to use Netscaler for HTTPS /HTTP for enable u website on our Linux:
Process is similar but an additional step is to install the SSL certificate on the Netscaler instead. This is SSL offload and there will be more controlled and features load balancing because Netscaler will be able to decrypt the SSL traffic and inspect the data.

Appendix B:

Netscaler / Native linux LB -

How did you develop native linux LB ? be as detailed as possible ? How would we implement HA design

We tested a self-built load balancing solution from the open source community which was a combination of PEN, pound, UCARP (equivalent of VRRP in open world), and HTTP check TCP check perl scripts from the Nagios world, which worked well in the in VMware image for our TCP listener service hosted in zLinux, and are currently working to compile the same in system Z as all of these package sources are available for free.

UCARP is the open source world IP failover protocol like VRRP which Cisco claims the ownership for.

for native LB scenario ?

Appendix C:

SAMBA -

what development need or business need initiated the requirement to implement Samba ?

Files had to be shared between Windows and Linux. Using NFS client on Windows was not efficient because of the limited ACL and permission related issues and the client software license was to be purchased and did not support multiple NFS shares on the same drive letter. Thus Samba turned out to be a better option because of its ease of use and simplicity and no coding changes required, and being free.

How was it procedural wise to implement ?

Implementation is very easy - install some packages, configure shares, set up user ID mapping. Setup passwords.

Appendix D:

Driver 93 problem-

Driver 93 update required that the Linux kernel version should be greater than version version 2.6.32.29-0.3.1. most of our images were patched 3 month ago and had kernel 2.6.32.27-0.2-default and thus did not meet the requirement (except our production environemnt luckily -kernel 2.6.32.59-0.3-default). because of the new feature called QIOASSIST(Queued I/O Assist) on Hipersocket in driver 93, the linux kernels using hipersocket panic'ed because IBM decided to keep it enabled by default. Many of our guests including our infrastructure servers like Our SMT (subscription management) server started kernel panic abruptly.

The solution was to repatch the systems to the latest, those which are failing but to be able to patch SMT was required and SMT itself was failing. Fortunately there was a 'CP QIOASSIST OFF' option at the guest level by using 'NOQIOASSIST' in z/VM guest level which we did on the SMT so it was available for other guests to use for patching. And we also had to do this on every guest we wanted to patch because we did not want the guest to panic while it was being patched.

Kernel panic was happening at the time of hyper socket initialization phase, so the guest that were not using hyper socket were safe but we use it on almost every guest.

Hyper socket was a different network segment and there was no failover option thus.